CENTRAL CIRCULATION AND BOOKSTACKS

The person borrowing this material is responsible for its renewal or return before the **Latest Date** stamped below. **You may be charged a minimum fee of $75.00 for each non-returned or lost item.**

Theft, mutilation, or defacement of library materials can be causes for student disciplinary action. All materials owned by the University of Illinois Library are the property of the State of Illinois and are protected by Article 16B of Illinois Criminal Law and Procedure.

TO RENEW, CALL (217) 333-8400.

University of Illinois Library at Urbana-Champaign

APR 25 2[illegible]

When renewing by phone, write new due date below previous due date.

L162

Digitized by the Internet Archive
in 2013

http://archive.org/details/monitorcommandsy912conn

UIUCDCS-R-78-912

Math

UILU-ENG 78 1705

510.84
I.L6r
no.912
Cop.2

The Monitor Command System User's Manual

by

Richard L. Conn

February 1978

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

The Library of the

MAR 27 1978

University of Illinois
at Urbana-Champaign

UIUCDCS-R-78-912

The Monitor Command System User's Manual

by

Richard L. Conn

February 1978

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Acknowledgments

I would like to thank Professors Roy H. Campbell and Michael Faiman for their encouragement and support, Edmund J. Dewan for his assistance in document preparation, and the secretaries and staff of the Department of Computer Science and the Computing Services Office of the University of Illinois for their general support. I would also like to give special thanks to Professor Alfred C. Weaver, now of the University of Virginia, for his encouragement, support, and the many hours of reading, editing, and re-reading he has devoted to this paper.

THE MONITOR COMMAND SYSTEM USERS' MANUAL

TABLE of CONTENTS

Chapter 1: Introduction to MCS 1
- 1 Overview of MCS 1
- 2 Elementary functions 3
- 3 Basic definitions 4

Chapter 2: The MCS Control Programs 5
- 1 Overview 5
- 2 MCS Line Editor 5
- 3 Comments 6
- 4 Line Feed Character 7
- 5 Edit control characters 7
- 6 Carriage Return 8
- 7 The MCS Command and Argument Parser 9
- 8 Argument Evaluator 9
- 9 Error Correction 10

Chapter 3: The Basic MCS Commands 11
- 1 Overview 11
- 2 Commands in detail 12

Chapter 4: Alternate Command Sets 17
- 1 Overview 17
- 2 Format 18
- 3 An ACS Example -- A Debug Package 20

Chapter 5: The MUMS Paper Tape Format 22
- 1 Overview 22
- 2 Checksum errors 23
- 3 Load Blocks 23

Chapter 6: MCS Utility Subroutines 24
- 1 MCS Utility Subroutines 24
- 2 System Subroutines 27
- 3 Buffer Addresses 28

CHAPTER 1

INTRODUCTION TO MCS

1 Overview of MCS

The Monitor Command System (MCS) is an operating system monitor and utility subprogram package. While its design is machine-independent, its first implementation was on an Intel 8080. This software package provides a set of basic operating system routines which will be useful in any 8080-based system, from a simple stand-alone system (microprocessor, memory, and teletype) to a complete microcomputer (e.g., IMSAI, ALTAIR) with peripherals (e.g., TTY, CRT, audio cassette, floppy disk). Memory requirements are 1K (1024 8-bit bytes) of read-only memory for the monitor itself (i.e., one Intel 2708 PROM) and 512 bytes of RAM.

MCS is designed to supply the user with the common functions necessary to load and debug a program and with the utility subroutines necessary to simplify the programming task. MCS can eliminate or reduce the problem of constantly re-writing the most common utility routines required to perform the elementary functions (e.g., input/output, number base conversion). Additionally, MCS allows the user to work with two sets of monitor commands -- the primary, or default, set of commands and an alternate command set. By loading an alternate command set the user

may control and debug his program through a specialized set of commands for his specific application.

There are basically three types of functions provided by MCS. First, it provides the 8080 user with a host of monitor commands through which he can enter, debug, and run his program; second, it provides the user with the ability to specify his own specialized set of commands which he may access through the monitor; third, it provides the user with a basic library of utility routines which he may find useful in many applications.

MCS was designed around four a priori constraints:

1. MCS should not be larger than 1024 bytes. This was imposed to reduce the amount of memory overhead required.

2. MCS should not reside in low memory (i.e., hexadecimal locations 0 through 3FF), since these addresses cover the restart area, and the user may wish to use these locations for his own purposes.

3. MCS should be autonomous, i.e., should not affect the environment of the user's program. When MCS is used for debugging, it is transparent to the user's program in the sense that the user register contents are unaffected by entering MCS.

4. MCS should be modular and structured to facilitate external access to all major subprograms (such as the examine and deposit routines).

MCS requires a small amount of scratchpad memory (512 bytes of random-access memory located at hexadecimal locations EE00-EFFF) for its internal use. Aside from this one RAM data area, MCS is a self-contained system and is ideally suited to reside in a read-only memory.

2 Elementary functions

MCS provides the following basic monitor functions:

1. examine and modify bytes of memory;

2. load programs and data into memory via paper tape;

3. execute a program located in memory;

4. debug a program by use of breakpoints. The user may

    1. set a breakpoint in the user program which causes control to return to the monitor when it is reached during program execution;

    2. examine the contents of all registers when the breakpoint is reached;

    3. alter the user's registers while in MCS and execute the user's program with new values in the registers;

5. invoke an alternate set of commands which may be used like the primary set of MCS commands, but which supply the user with specialized functions.

3 Basic definitions

For the purpose of this paper, the following terms are defined:

1. TELETYPE, or TTY -- any I/O device connected to the system in use; this device is preferably an ASR33 teletype, since the ASR33 has a paper tape reader. Although the TTY may in fact be any I/O device (e.g., a CRT), if it does not have a paper tape reader the Load command (described later) is not operative.

2. <CR>, <LF>, <ESC>, <DEL> -- these symbols refer to specific keys on the TTY. <CR> is the carriage return, or return, key; <LF> is the line feed key; <ESC> is the escape, or altmode, key; and <DEL> is the delete, or rubout, key. The functions of these keys are discussed later.

3. N8 and N16 -- these symbols represent numeric quantities. N8 refers to an 8-bit quantity and N16 refers to a 16-bit quantity. N8 is not a subset of N16; i.e., N8 can have a value from hexadecimal 0 to FF while N16 can have a value from hexadecimal 100 to FFFF.

4. B -- blank or space (<SP>)

5. *,+ -- these symbols, concatenated to a parenthesized expression or other symbol, indicate multiplicity; '*' indicates zero or more, '+' indicates one or more. Hence, B* indicates zero or more blanks, B+ indicates one or more blanks, (B+ N8)* indicates zero or more strings concatenated, each string containing one or more blanks and one N8 (example: ' E3 E4 E5' would be a valid derivative of this expression).

6. ? -- this symbol indicates that the use of the immediately preceding symbol is optional. Hence, N8? means that the N8 is optional, (B+ N8)? means that (B+ N8) is optional.

CHAPTER 2

THE MCS CONTROL PROGRAMS

1 Overview

The Monitor Command System is run by three user-accessable control programs: (1) INLINE, a line editor, (2) GETIN, a command and argument parser, and (3) PTEST, a "case statement" which calls the appropriate subroutine to execute the desired function. This chapter discusses INLINE and GETIN in detail, since they are very useful routines which the user may wish to call from his own program.

It should be noted that these subroutines rely upon a number of utility subroutines, and these, too, are user-accessable. All MCS subroutines are discussed in chapter six.

2 MCS Line Editor

The MCS Line Editor is used to enter and edit all commands to MCS from a teletype (TTY) or TTY-like device. Editing may be accomplished while the command is being typed; entry and edit of the command is terminated by pressing the carriage return (<CR>), or return, key.

There are basically two types of characters recognized and acted upon by the MCS Line Editor -- normal characters and control characters. Normal characters are stored in an input buffer until a <CR> is typed; they are then processed and the results returned to the calling program (MCS). Control characters are special characters recognized by the line editor, and their respective functions are executed immediately after the typed command is recognized.

The line editor enables the user to type his command to the monitor on his TTY. It inputs each character he types, examines it to determine whether it is a control character, executes it immediately if it is and saves it if it is not, and echoes the character onto the TTY. All characters are significant to the MCS Line Editor. When the editor is exited, the input buffer (IBUF) contains exactly those characters typed by the user except for the control characters. The only restriction imposed upon command line input is that the total number of characters may not exceed 256. If it does, wrap around occurs and the remaining characters input overwrite the first characters typed in.

3 Comments

A comment is formally defined as a string of characters which begins with either an exclamation mark (!) or a semicolon (;) and which ends with a <CR>. Any non-<CR> character, printable or non-printable, may be placed within a comment.

A comment may be placed anywhere within a line. If placed at the beginning of a line, the entire line is processed by MCS as a comment; if placed after a command, the command is processed as though the comment was not present. The MCS Line Editor itself does not define comments and it handles them as part of the input string. It is not until MCS gains control from the Line Editor that the comment is defined and processed.

4 Line Feed Character

The Line Feed (<LF>) is a control character processed by the MCS Line Editor. This character enables the user to continue typing his command on the next physical line of the TTY. It is executed by hitting the Line Feed key and it echoes onto the TTY as a <CR> <LF> pair, thereby returning the carriage to the left margin and advancing the paper one line. <LF> is not entered into the input buffer; hence, as many line-feeds as desired may be placed anywhere in the input line without altering the interpretation and processing of the line.

5 Edit control characters

The MCS Line Editor responds to three editing control characters: the escape (<ESC>), or altmode, key, the delete (<DEL>), or rubout, key, and the CTRL-H, or backspace, key. These keys provide very simple and useful editing features.

1. Escape

The <ESC> key deletes the entire line and restarts the input of the command. It echoes as a dollar sign ($) and a <CR> <LF> pair. After the <ESC> key is input the user may then type as though he just started to enter a command. The <ESC> key may be typed as many times as desired.

2. Delete

The <DEL> key deletes the last character typed. It echoes as a backslash (\) followed by the character deleted. If the <DEL> key is typed again, the next preceding character is echoed and deleted; this action repeats each time the <DEL> key is pressed. If the user attempts to delete the undefined character preceding the first character in the line, an <ESC> is automatically processed.

After the desired characters have been deleted, the deletion procedure is terminated by typing the next character desired; this character is echoed as a backslash followed by that character. Entry of the line may then continue as it normally would and further deletions may be made as desired.

3. Control-H

The CTRL-H key performs the same function as the <DEL> key, except that it is oriented toward soft-copy terminals such as CRT's. It echoes as a backspace and deletes the previous character in the line when it is typed. An <ESC> is processed if the user tries to delete the undefined character preceding the first character in the line.

6 Carriage Return

As mentioned before, a <CR> terminates the entry and edit of a line. When a <CR> is typed, an end-of-line (EOL) character is placed after the last character in the input buffer. The EOL character is a hexadecimal FF.

7 The MCS Command and Argument Parser

The MCS command and argument parser, GETIN, is a subroutine which uses INL1NE to input a string of characters which represent a command (see chapter three for a formal definition of a command). It then extracts the one-letter command, if any, and builds a buffer which contains the binary values of the arguments.

Upon return to the calling program, the command is in the A register and the D&E register pair contains the value of the first argument. If there is no first argument, D&E contain the value of the default pointer contained in the B&C register pair (i.e., D&E=B&C). Also, all the binary (N16) values of all arguments are contained in the buffer VCNT, in which the first byte is the number of arguments and the succeeding bytes are the N16 values stored as low order 1, high order 1, low order 2, high order 2, etc. The command is also stored in the one-byte buffer CMND.

GETIN is not an autonomous routine; all registers are affected by it. If the user has some important information in the registers, he should save it, preferably with the Push and Pop instructions, before calling GETIN.

8 Argument Evaluator

GETIN uses the subroutine GET to evaluate arguments in the command string and this subroutine is available to the user. On input, the H&L register pair point to the first character of an argument character string. GET then starts converting characters to hexadecimal values in the D&E register pair until an invalid hexadecimal digit (i.e., a character which is not 0-9 or A-F) is encountered. Upon return, the B&C register pair is not changed, D&E contain the N16 value of the last four hexadecimal characters of the argument (if there are more than four characters in the argument), and H&L point to the invalid hexadecimal character which stopped the processing.

9 Error Correction

Upon return from GET, GETIN stores the D&E register pair in the correct locations in VCNT, with E being stored first and D second (low order, high order). It should be noted that this facility is somewhat useful in error correction in which, rather than correcting a mistake in an N8 argument, the user may simply type the proper digits and they will be loaded into E by GET, and the improper digits will be loaded into D.

Care should be taken in using this error-correction facility. If the argument is of type N16, as is the first argument of the D Command, this facility must be used with the knowledge that the last four hexadecimal digits must represent the desired value.

For instance, if the user typed 'D EC' when he wanted 'D DC', he must type '00DC' to correct the error using this facility. As a result, the entire string will look like 'D EC00DC', and will be processed as 'D 00DC'. On the other hand, to correct 'D DC DD' to make the last value 'ED', all that is necessary is to type 'ED', rather than '00ED', since the 2-to-N arguments of the D Command are of type N8. Hence, 'D DC DDED' will be processed as 'D DC ED'.

The user must know at all times the types of arguments a command calls for in order to use this error-correction facility to his advantage.

CHAPTER 3

The Basic MCS Commands

1 Overview

The Monitor Command System provides the user with a small but very powerful set of monitor commands. These commands allow the user to examine and modify memory, run programs stored in memory, and invoke an alternate command set which gives the user an extended set of specialized commands. This chapter will discuss these commands in detail and provide the user with a working knowledge of how to use them.

All commands have the same general format, viz. a prompt followed by the one-letter command and its arguments, if any. The prompt is printed by MCS, and the user types the command and its arguments. The Extended-BNF notation for an MCS command is:

```
Command : ('.' ! '>') B* C B* (N16 B+)* Comment
```

where:

```
B : "Blank"

C : ('A' ! 'B' ! 'D' ! 'E' ! 'L' ! 'R' ! 'S')

Comment : (';' ! '!')  Charset

Charset : "any character in the ASCII character set"
```

The commands to be discussed in this chapter are:

A -- Invoke the alternate command set. Transfer control to the alternate command set from the primary (normal) command set.

B -- Set a breakpoint. Set a breakpoint anywhere in memory.

D -- Deposit. Deposit one or more values into memory.

E -- Examine. Examine the contents of memory.

L -- Load paper tape. Load memory with programs or data stored on paper tape in MUMS format (see chapter five).

R -- Run. Run a program located in memory.

S -- Set default pointer. Set or examine the default pointer, viz. sometimes used in conjunction with the other commands.

2 Commands in detail

The basic MCS commands are presented below. The format of the presentation is the name of the command, the format for using the command, and one or more paragraphs which describe the command.

1. A, or ALTERNATE, command. Format: A (N16)*

The A Command is used to invoke an alternate command set, viz. a group of commands which are used exactly like the Primary Command Set (PCS) described in this report. These Alternate Commands provide the user with another set of functions he can invoke through the monitor.

There are two types of Alternate Command Sets: Unary and Multiple. A Unary Alternate Command Set (UACS) is a command set which consists of only one command; a Multiple Alternate Command Set (MACS) is a command set which consists of several commands. If the command set is a UACS, any arguments the command requires are provided through the A command in the PCS. For instance, if a UACS requires an address as an argument, the A command would be of the form 'A N16'. If the command set is a MACS, the A command should not have an argument, since all the required arguments are supplied in the individual alternate commands given by the user.

When a MACS is invoked, a '>' is given as a prompt. The user may then give the commands peculiar to the particular MACS he is using. One command is common to all MACSs: the X, or eXit, command. This command exits the MACS and returns to MCS.

To use any ACS, the particular ACS the user desires must be loaded into memory. This is done, usually, by acquiring the paper tape containing the desired ACS and using the L command to load it. After the load, the ACS is ready to be used.

2. B, or BREAKPOINT, command. Format: B (N16)?

The B Command is used to set a breakpoint in memory. A breakpoint is a location which, when the machine code located there is executed by the microprocessor, transfers control back to the user through MCS. When a breakpoint is reached, MCS saves all register and stack pointer values. In this way, the breakpoint command

provides an excellent debugging tool. Executing a breakpoint preserves the environment of the program while also allowing the user to examine the registers and use the commands available through MCS.

The breakpoint itself is a subroutine call statement which requires three bytes of memory. Hence, a warning should be made: When setting a breakpoint, make sure the program cannot transfer control to bytes two or three of the breakpoint. If this happens, the microprocessor will try to execute the address, or part of the address, given in the breakpoint as an instruction, and the user's program could be destroyed.

When a breakpoint is executed, it is reset, i.e., the breakpoint is replaced by the bytes of the user's program it originally overwrote, so the user's program is restored to its original form.

The B Command may or may not have an argument. If an argument is present, that argument is the address of the breakpoint; if there is no argument, the address of the breakpoint is the default pointer. The default pointer is set by the S command and modified by the D, E, and R commands. The A Command may also modify it, depending on the specific functions executed.

3. D, or DEPOSIT, command. Format: D (N16)? (N8)*

The D Command is used to deposit values into memory. Any number of values may be deposited by one D Command, provided that the input buffer does not overflow, and the starting address of the deposit may be either set by the D Command or the default pointer.

If the D is followed by a 16-bit quantity, i.e., a number which requires two bytes of storage, that quantity is the starting address of the deposit. Hence, any address in the first 256-byte page of memory cannot be specified by the D Command, but any other address can. If the user wishes to deposit into the first page, he must use the S or E commands to set the default pointer, and then use the D command with just values as arguments.

The values are deposited starting at the starting address and they are deposited in sequential locations of memory. When completed, the default pointer points to the byte following the last deposited value, so that another D Command will continue depositing where the last one quit.

4. E, or EXAMINE, command. Format: E (N8 ! N16)?

The E Command is used to examine the contents of memory. If it has an argument, the examination starts at the address specified; if it has no argument, the examination starts at the default pointer.

After the command is typed, MCS will print the address followed by its content and a question mark (?) as a prompt. In response to the prompt, the user may type 'F' (Forward) to examine the next location, B (Backward) to examine the previous location, R (Run) to run at the current address examined, or X (eXit) to return to MCS. If F or B are typed the appropriate address and content are displayed with prompt and the user may continue as before with another F, B, R, or X command.

Upon exit from the examination, the default pointer is set to the last address examined.

5. L, or LOAD, command. Format: L

The L Command is used to load memory with programs or data coded on paper tape in MUMS format (see chapter five). MCS will turn on the paper tape reader, assuming an ASR33 Teletype, and load the contents of the tape into memory. When finished, it will turn off the paper tape reader and prompt the user with the message '**?', which indicates that the load is complete. The user is then to type an 'X' to eXit the loader and return to MCS.

6. R, or RUN, command. Format: R (N8 ! N16)?

The R command runs the program which starts at the address specified in the command or implied by the default pointer. If no argument is present the address is supplied by the default pointer. The registers and stack pointer are restored from the previous run, if there was one, and control is transferred entirely to the program to be run. The default pointer value is lost, and if a breakpoint is reached or the program transfers control back to MCS, the value of the default pointer will be indefinite, as it is upon entry into MCS initially.

7. S, or SET, command. Format: S (N8 ! N16)?

The S Command is used to set and examine the default pointer. If an argument is present, the default pointer is set to the value of that argument; if there is no argument, the value of the default pointer is displayed.

CHAPTER 4

ALTERNATE COMMAND SETS

1 Overview

An Alternate Command Set (ACS) is a second set of commands the user may access through MCS. Generally loaded by the paper tape loader routine, an Alternate Command Set is a very useful tool in specific applications.

After the user has loaded an alternate command set, he may then use the alternate commands by typing the A Command in MCS. If the ACS is a UACS, the user types the A Command as the letter 'A' followed by the required arguments, if any; if it is a MACS, the user simply types the letter 'A'. In the case of a UACS, the command will be executed and control will then return to MCS; in the case of a MACS, MCS will prompt the user with a '>', after which the user may then type the specific alternate command desired, and control will return to the alternate command section of MCS (a '>' will be given as the prompt) when the alternate command invoked is finished. Generally, the user exits an ACS and returns to MCS via the 'X', for eXit, Command (which consist of only the letter 'X').

2 Format

An ACS has a relatively simple format. It consists of a table of commands starting at hexadecimal EC00. If the ACS is a UACS, the commad table is of the form:

```
EC00:  0

EC01:  <low order byte>

EC02:  <high order byte>    ,
```

where the low order and high order bytes specify the address of the routine which executes the command. If the ACS is a MACS, the command table is of the form:

```
EC00:  <ASCII value of  first command letter>

EC01:  <low order byte>

EC02:  <high order byte>

EC03:  <ASCII value of second command letter>

 .         .

 .         .

 .         .

ECxx:  0      (end of table indicator)
```

The order (numeric, alphabetic) of the commands in the table is immaterial. The only requirement is that the programmer not specify two commands with the same letter. If this is done, the second command in the table will never be executed.

The ACS routine itself may be anything the programmer wishes provided that the stack pointer is the same at the start of the routine

as at the end. This is necessary because MCS pushes onto the MCS stack the return address which the ACS command should return; to return to the Alternate Command section of MCS, the programmer need only terminate his program with a 'RET' instruction. For example,

```
E800:  PUSH H

E801:  POP  H

E802:  RET

EC00:  44 (HEX)      ; LETTER 'D'

EC01:  00

EC02:  E8

EC03:  45 (HEX)      ; LETTER 'E'

EC04:  00

EC05:  C0

EC06:  00         ; 0 to terminate table
```

illustrates an ACS which is viable but does nothing interesting.

An alternate command is processed by MCS like a primary command; i.e., the user enters the command through the line editor, and the command buffer (CMND) and variable buffer (VCNT) are built. Hence, the ACS programmer may use the arguments in VCNT in his program.

To the user, the distinction between the command sets is obvious -- if he is in the primary command set of MCS, MCS prompts him with a '.'; if he is in an ACS, MCS prompts him with a '>'. Hence, he knows by the prompt which set of commands to use.

Finally, the X command is generally used by MCS to exit an ACS. If there is an X command defined in the MCS, this overrides the X command defined by MCS and the ACS programmer should provide for an exit to MCS if he defines his own type of exit command.

3 An ACS Example -- A Debug Package

The Appendix to this paper lists an example of an ACS designed to aid the user in debugging his programs. The commands executed by this debug package are:

1. E B* (N16 ! N16 B+ N16)? -- This command is used to examine either one memory location or a block of memory. The dump is displayed 20 lines at a time, and the user is prompted with 'CONT?', to which he responds with 'Y' to continue or anything else to stop.

2. D B* N16 (B+ N8)+ -- This command is used to deposit a string of bytes into memory. N16 is the address to start the deposit, and each N8 is a byte to be deposited.

3. P B* N16 B+ N16 -- This command dumps a block of memory onto paper tape in MUMS format.

4. T B* N16 B+ N16 B+ N16 -- This command transfers a block of memory whose starting address is the first value and whose ending address is the second value to the memory location starting at the third address.

5. F B* N16 B+ N16 (B+ N8)+ -- This command finds all occurrances of the string of bytes specified by the N8's in the block bounded by the starting and ending addresses given.

6. S (B+ N8)+ -- This command defines a string of characters to be used in substitution with the W Command.

7. W B* N16 -- This command defines the starting address of a substitution. The string of bytes specified by the S command is moved into consecutive memory locations beginning at the address specified in the W Command.

8. L -- The link command. This command engages the link program, which provides downloading facilities from an external source. While in the link, a CTRL-R returns the user to MCS and a CTRL-L loads memory with data which comes in from the external source in MUMS paper tape format.

CHAPTER 5

THE MUMS PAPER TAPE FORMAT

1 Overview

The paper tape format consists of an arbitrary number of load blocks separated by strings of '@' symbols. The load block is defined as one unit of memory loaded through paper tape (or any other medium). It has the following general form in the MUMS vernacular:

```
('@'* '#' <hex digit><hex digit><hex digit><hex digit>
                (<hex digit><hex digit>)*
        '&' <hex digit><hex digit> '$'? '@'*)+
```

When the paper tape loader in MCS receives a '#' it zeroes the local checksum word, decodes the next four digits as an address, and then loads data into consecutive memory locations starting at this address until the '&' symbol is received. The loader then compares the following two digits with the checksum value computed and signals the checksum error message if they do not match. The loader also ignores all '@' symbols and <CR> <LF>s which appear in the paper tape.

2 Checksum errors

Upon encountering a checksum error the tape may be restarted at the beginning of the load block which was in error, rather than at the beginning of the tape; it is assumed that the mis-read digit was not an address digit. This feature will facilitate the loading of large programs.

3 Load Blocks

A load block may have a maximum length of 512 object digits (256 bytes). The current load block is terminated whenever the location counter is redefined in the source program, whenever the maximum block length is exceeded, or whenever the end of the program has been reached.

CHAPTER 6

MCS UTILITY SUBROUTINES

1 MCS Utility Subroutines

MCS provides a host of utility subroutines accessable by the user. The following is a list of these routines, and information is presented in the following format: name of routine; brief description of what the routine does.

1. BSLSH; prints a backslash on the principal I/O device

2. CATH; converts the ASCII character in the A register into its corresponding hexadecimal equivalent. If it is an invalid hexadecimal digit (i.e., not 0-9, A-F), a <SP> (ASCII value of hexadecimal 20) is returned in the A register.

3. CBC; the B&C register pair is decremented, and the Carry bit is set to 0 if the result is 0; else, the Carry bit is set to 1

4. CHTA; converts the low nybble (4 bits) of the A register to the ASCII value of this hexadecimal digit

5. CRLF; prints a <CR> <LF> on the principal I/O device

6. EN; exchanges the nybbles of the A register

7. GET; obtains the value of the string of hexadecimal characters pointed to by the H&L register pair in the D&E register pair (discussed in chapter two)

8. GETIN; this is the command and argument parser (discussed in chapter two)

9. INLINE; this is the MCS Line Editor (discussed in chapter two)

10. INPUT; this routine obtains one character from the principal I/O device

11. LF; this routine prints a <LF> on the principal I/O device

12. M3DH; this routine moves three bytes, the first of which is pointed to by the D&E register pair, to the three bytes which start at the location pointed to by the H&L register pair

13. MCS; this is the starting point of MCS

14. MLOOP; same as M3DH, but the number of bytes specified in the B register is moved

15. OUTPUT; this routine prints the character stored in the A register onto the principal I/O device

16. PA2C; this routine displays the contents of the A register as two hexadecimal digits on the principal I/O device

17. PCHAR; this routine displays the character pointed to by its return address on the principal I/O device and returns to the byte following this character. It is generally used as follows:

```
CALL PCHAR
DB   'A'
```

18. PCHAR2; same as PCHAR, but the following two bytes are displayed

19. PEM; same as PCHAR2, but '**' is displayed before the two characters

20. PHL; prints the H&L register pair as a string of four hexadecimal digits

21. PIN; inputs a character from the Paper Tape Reader

22. POUT; outputs a character in the A register to the Paper Tape Punch

23. PRBL; prints a <SP> on the principal I/O device

24. PRINT; prints the string of characters pointed to by the return address until a binary 0 is encountered

2 System Subroutines

The following is a list of the system subroutines which the user may wish to access when he develops his own ACSs. All of these routines except DEPOS depend upon the D&E register pair for the address argument. All registers should be considered volatile and the first return address on the stack is thrown away except in the PTEST and RUN subroutines.

1. ALTER; this subroutine invokes the ACS. Generally, the user will not use this unless he wishes to have an ACS that is the subset of his ACS, but upon return to MCS, the second ACS will be the active one.

2. BPRET; this is the location to which a breakpoint returns to MCS. After the housekeeping is done, control is returned to MCS.

3. BREAK; this is the routine which sets a breakpoint.

4. DEPOS; this routine does the multiple-argument deposit.

5. EXAM; this routine executes the E Command.

6. PTL; this routine loads paper tape in MUMS format.

7. RUN; this routine executes the R Command.

3 Buffer Addresses

The following is a description and list of the buffers used by MCS in RAM.

1. BPI; EF00; 3; this buffer is used to store the three bytes replaced by the breakpoint when a breakpoint is set. After the breakpoint is reached, these bytes are restored in the user's program but this buffer still contains them.

2. CMND; EF06; 1; this buffer contains the command that is extracted by GETIN.

3. IBUF; EE00; 256; this buffer is the input buffer generated by INLINE.

4. RSAV; EFF8; 8; this buffer is used to save the values of the registers. The order of the save is (from EFF8 to EFFF): PSW, A, C, B, E, D, L, H.

5. SPSAV; EFF6; 2; this buffer is used to save the value of the stack pointer (low order, high order).

6. VCNT; EF07; 122; this buffer is used to store the number of values in the argument string as its first byte and the values themselves (as low order, high order) in each pair of bytes thereafter.

7. VPTR; EF04; 2; this buffer is used by GETIN to point to the next byte in VCNT to store a value.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-78-912	2.	3. Recipient's Accession No.
4. Title and Subtitle The Monitor Command System User's Manual			5. Report Date February 1978 6.
7. Author(s) Richard L. Conn			8. Performing Organization Rept. No. UIUCDCS-R-78-912
9. Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No.
12. Sponsoring Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801			13. Type of Report & Period Covered Technical 14.

15. Supplementary Notes

16. Abstracts

The Monitor Command System (MCS) is a 1K-byte monitor. It is designed to be system-independent, and it has been implemented on an 8080-based microcomputer system. It provides a number of system utilities, such as input/output routines, and a set of user commands which provide memory manipulation, program loading, and program debugging facilities.

17. Key Words and Document Analysis. 17a. Descriptors

Monitor Command System

17b. Identifiers/Open-Ended Terms

microprocessor operating system
microprocessor monitor system

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
RELEASE UNLIMITED	UNCLASSIFIED	31
	20. Security Class (This Page) UNCLASSIFIED	22. Price

BOUND BY
HM

UNIVERSITY OF ILLINOIS-URBANA
3 0112 047353930